Best Viewpoint Tracking for Camera Mounted on Robotic Arm with Dynamic Obstacles

Best Viewpoint Tracking for Camera Mounted on robot Arm with Dynamic Obstacles

Christos Maniatis Marcelo Saval-Calvo* Radim Tyleček
Robert B. Fisher
School of Informatics, University of Edinburgh, cmaniatis,rtylecek,rbf@inf.ed.ac.uk.
*Dept. Computer Technology, University of Alicante, msaval@dtic.ua.es.

Abstract

The problem of finding a next best viewpoint for 3D modeling or scene mapping has been explored in computer vision over the last decade. This paper tackles a similar problem, but with different characteristics. It proposes a method for dynamic next best viewpoint recovery of a target point while avoiding possible occlusions. Since the environment can change, the method has to iteratively find the next best view with a global understanding of the free and occupied parts.

We model the problem as a set of possible viewpoints which correspond to the centers of the facets of a virtual tessellated hemisphere covering the scene. Taking into account occlusions, distances between current and future viewpoints, quality of the viewpoint and joint constraints (robot arm joint distances or limits), we evaluate the next best viewpoint. The proposal has been evaluated on 8 different scenarios with different occlusions and a short 3D video sequence to validate its dynamic performance.

1. Introduction

There are multiple scenarios in computer vision when it is necessary to find a proper location for a sensor in dynamic environments to perceive a target point such as assembly lines, surveillance, medical operation recording [2]. For example, one might want to have a video record of an operation, e.g. to demonstrate that the doctor followed best practice. Given the changing positions of the doctors' hands, limbs and tools, the relative position of the camera will also need to change dynamically to maximize viewability and also avoid occlusions.

In this paper we will focus on dynamic viewpoint selection with a moving sensor along with static and dynamic obstacles occluding a target point. The paper assumes a camera held by a robot in the configuration seen in Fig. 2. A similar problem is the Next Best View (NBV) [6, 11] where authors propose a method for planning of the camera positions to perform a 3D reconstruction of a given object. The problem addressed here considers how to maintain an optimal viewpoint, as contrasted with the traditional NBV problem, which aims to maximize total viewing. The optimality condition includes both visibility and camera motion terms. In our approach we will use the idea of the voxel map described in these papers that address NBV in order to reconstruct the scene from a set of 3D points.

This paper proposes a novel method for tracking the best viewpoint based on a 3D representation of the scene. The most important aspect of the algorithm is to handle both dynamic and static elements that occlude the target point. Our algorithm has to estimate the next best viewpoint and to achieve that a partitioning of the space is proposed as a tessellated hemisphere, where the center of every face is a possible camera position or viewpoint (hereafter called 'view'). The adequacy of each position in terms of occlusion, angle of perception and distance from the previous position will be defined using an objective function.

The main contributions of this paper are twofold:

- Introduction of a new type of problem: the Dynamic Next Best View (DNBV), where dynamic occlusions need to be overcome.
- An algorithm that solves the problem of DNBV.

In the following sections we will give an overview of the related work and introduce a description of the observed scene. Subsequently we will describe our optimization based solution and show experiments on real data.

2. Related work

The problem considered here is to find the best positions of the sensor to perceive a target point in a dynamic environment. We assume that the target to be observed remains stationary, but there are moving objects that occlude the target from different viewpoints at different times. Hence, there

are two main aspects to take into account: best view selection and tracking of the target location.

Similar problems related to the former aspect have been tackled. Finding the best viewpoint was coined in the literature as Next Best View (NBV) [8]. The NBV problem arises in the construction of a 3D model of an object. The aim is to find the best positions of the camera to perceive all the parts of the object [6, 11, 10, 5].

Morooka et al. [8] proposed an on-line algorithm that chooses the NBV based on an already obtained partial model. It uses an objective function that takes into account the possibility of merging new data, local shape changes, control distribution and registration accuracy. In [11] the NBV is defined with a camera pose which simultaneously allows good registration, elimination of occlusion plane areas and observation of unseen areas.

The problem of NBV is still a challenging problem to reduce the number of views needed to capture the whole object. Singh et al. [12] uses a labeling of the object in levels of perception quality given by the angle between camera and the object and ray tracing techniques. With this information, mean-shift clustering is applied and the cluster with the highest value is chosen as the next best view. Other approaches use contours to calculate the unseen parts [7]. Vasquez-Gomez et al. [13] used a two stage system that improves the quality of the modeling by predicting a next-best-view and evaluating a set of neighbor views, eventually selecting the best among all of them.

Our problem includes dynamic elements in the scenario, so we have to deal with moving occlusions. Zhang et al. [14] use the information of the occluding element and assigns the next best view by maximizing the perception of the surface out of the occluded region. This solution is not applicable to our problem since our environment is dynamic so the occlusion changes over time.

A second important aspect is to track the best viewpoint. Since the environment is dynamic, new occlusions can appear. Moreover, we are in a real environment where the camera movement has to be taken into account. We want to minimize the distance of displacement between viewpoints to increase image stability while maximizing the view quality as well as avoiding occlusions.

The dynamic visual tracking problem is a completely different problem to NBV. A camera mounted on a moving or stationary base is used to track an object. In [9] the authors defined visual tracking in 2D of a single feature point as the translation (T_x, Ty) and rotation (R_z) with respect to the camera frame that keeps Ω_w stationary, where Ω_w is the area in the image plane where the target is projected. Many authors e.g. [9, 4] have addressed this problem using different approaches.

Papanikolopoulos et al. [9] combined visual tracking and control. In particular they use sum of square differences of the optical flow to compute the discrete displacement which is fed to either a PI/pole assigned controller or a Kalman filter to improve the state estimates. These estimates are then used to move the robot arm. According to [9], [4] predicted mathematically the position of the object's centroid, in order to visually track the object. Their algorithm could only work for slowly moving objects, since it had to compute the coordinates of the centroid.

In our case, we are not focused on tracking the occlusions themselves. Our goal is to track the free views. Some authors have studied planning methods to achieve best camera positions, also called camera planning. Dunn et al. [1] presented a proposal with a recursive path planner to perform NBV. However, the environment does not change in their case, whereas we have to re-estimate the new position in terms of new occlusions, distance between current and future point and viewpoint quality.

Our problem to dynamically find the next best view includes multiple variables: the view quality, current occlusions in a dynamic environment and distance minimization between the current viewpoint and the new one. To the best of our knowledge, there is no previous work combining these problems. Our paper presents a novel dynamic next best view camera planning algorithm including these three main variables.

3. Proposed solution

The key to our algorithm for Dynamic Next Best view (DNBV) is reconstructing the dynamic 3D scene with data captured from sensors that observe it as in Fig. 2. Here, four Kinect V2 sensors are used to capture activity in a workcell setting. The voxel map described in the Sec. 4.2 is used for this purpose. Then a constrained tessellated view sphere is initialized similar to the ones suggested in [6, 8], presented in Sec. 4.1. Next the voxel map is projected onto the tessellated sphere and faces that correspond to sets of unoccluded projected voxels are identified as candidate viewpoints. Then a new position for the robot arm can be predicted by maximizing an objective function that will take into account the occlusion, the viewpoint quality, and finally how far the sensor will have to move.

In our case there is a camera mounted on a robot arm. Similar ideas to the ones described above could also be found in [4, 9]. The above procedure is summarized in Fig. 1.

4. Initialization

This section will describe how to initialize the tessellated hemisphere and the occupancy grid. It also explains the indexing procedure that is used for the occupancy grid.

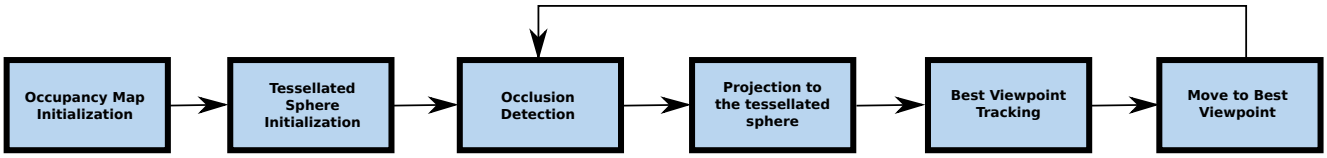


Figure 1: Scheme of the proposed method.

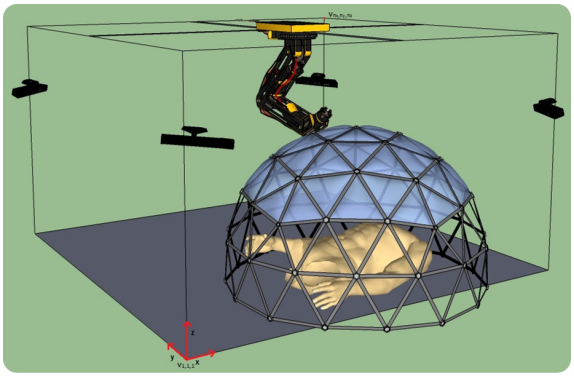


Figure 2: Depth sensor setup for tabletop setting.

4.1. Tessellated view sphere

As one task-based constraint is to avoid interfering with the people in the workcell, the camera is assumed to be at some distance from the target. Here, the surface of a sufficiently large hemisphere defines the 2 DOF space of camera positions (azimuth, elevation), assuming the camera always faces upward.

In general the space of possible views is continuous, but here we limit and discretize the space to the surface of a tessellated sphere, also referred as the dome in this paper. We restrict the movement of the robot arm such that camera mounted on the end effector moves on the surface. To obtain a discrete set of possible positions we tessellate the sphere, such that all the positions are equally likely to be chosen. In the literature there are many ways to tessellate a sphere.

Here, the set of potential viewing positions is defined by the face centers of a half icosahedron centered above the target in the workcell. The highest point of the dome is the north pole and the set of the lowest points is the equator. The facets lying on the equator are removed, since we will restrict attention to the upper area of the dome as described in the next section which presents the objective function.

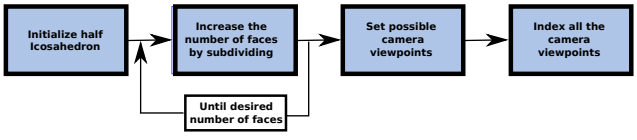


Figure 3: Scheme of tessellated hemisphere initialization.

The center of each facet is a possible camera position. With the initial dome there are only 10 possible camera positions. In order to generate more viewpoints, each face is subdivided into four equal pieces, by connecting the centers of the edges of each triangle. This is repeated until the desired viewpoint resolution is achieved. Indexing each camera position starts from one of the closest positions to the north pole by labeling it as cp_1 then in a spiral passing through all the unlabeled points until the equator is reached. The generation of the tessellated hemisphere is summarized in Fig. 3.

4.2. Voxel map

In the literature a common tool used in the implementation of NBV to identify occluded voxels is a voxel map. According to [6] the voxel map is a volumetric representation of the scanned space including the object we are interested in. Each voxel is a fixed volume cube whose size is pre-specified. The volume of each voxel is based on the width, height and depth of the scene. Then it is discretized to n_x, n_y and n_z points for x, y and z dimension respectively. The v_{xyz} identifies each voxel, then the occupancy

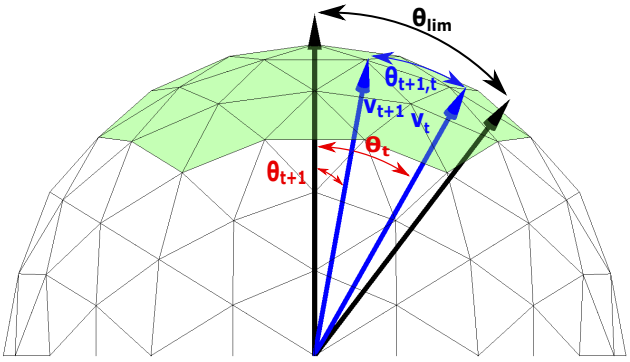


Figure 4: Tessellated hemisphere (dome) with two camera poses for the current (t) and next ($t+1$) state. The θ_{lim} is the maximum allowed angle, θ_t and θ_{t+1} are the angles that camera viewpoints at states $t, t+1$ make with the normal to the table and $\theta_{t+1,t}$ is the angle between viewpoints v_t and v_{t+1}.

grid is denoted as $X_t = \{v_{xyz}\}$, where $x = 1, \ldots, n_x$, $y = 1, \ldots, n_y$ and $z = 1, \ldots, n_z$.

Apart from identifying each voxel we also need a systematic labeling when assigning the points of the point-cloud to each voxel. Sanchiz et al. [11] describe a straightforward method to achieve that using the round-scale function. The labeling of each voxel varies from one author to the other but all are similar. Those labels could be discrete (ie. only one possible label) or probabilistic [3]. Here, the following discrete set of labels for v_{xyz} is used:

- *Empty*. These voxels are empty and are normally between the sensor and the surface of the scanned object
- *Seen*. These are normally the voxels that include the surface of the scanned scene, including any occluding objects. For the voxels with this labeling we also estimate quantities related to region quality as well as the surface normal.
- *Unseen*. The voxels that have not been scanned.

The seen voxels of X_t, represented with their center point, are then projected onto the tessellated sphere to determine target visibility and viewpoint occlusion. To do that, the tessellated dome is approximated with a hemisphere whose origin lies just above the target point (and so here contains only occluding voxels). For each voxel center v_{xyz} inside the dome the polar and azimuth angles ϕ and θ of its projection onto the hemisphere are computed. Hence given a radius its coordinates on the hemisphere can be calculated. Then for each projected point the geodesic distance to all dome vertices is computed and the three closest are found, which allows identification of the face i that the point projects to[1]. In total m_t^i denotes the number of such 3D points that project to face i. How these projected points are used is discussed in Sec. 5.2.

4.3. Instantiation of the occupancy grid

There are a number of methods for instantiating the values of an occupancy grid [3]. Irrespective of the sensor used, it is assumed that the data is a 3D point cloud $\{\vec{x}_i\}$ representing the scene. The occupancy grid is instantiated as *unseen* and then all voxels containing a 3D point are marked as *seen*. Finally, all voxels lying on the ray from the viewpoint center to the surface past the final *seen* voxel are marked as *empty*.

Note that as this is a dynamic problem. The occupancy grid is recalculated each time a new point cloud is available. This differs from the standard NBV algorithm which fuses the new point clouds instead of replacing them.

[1]There is probably a more efficient algorithm than computing all distances.

5. Objective function

This section introduces the objective function used to predict the next best view. The possible camera locations (Fig. 4) are described with a set of parameters: r is the radius of the dome, $\vec{n}$ is the normal of the workcell surface, $\vec{v}_b$ the direction of the view of the camera when on the surface of the dome at point b and a is the maximum allowed viewpoint angle.

To model the problem the current state t is described with the following parameters:

$$s_t = \{\vec{v}_t, \vec{j}_t, \vec{m}_t\}, \tag{1}$$

where $\vec{v}_t$ is the direction of the viewpoint (unit vector), $\vec{j}_t$ are the joint parameters of the robot arm and the $\vec{m}_t = \left\{m_t^i\right\}$ is a vector that counts the voxels occluding each each view.

The objective function is a linear combination of probability-like distributions that will account for visibility, occlusion level and distance that needs to be traveled by the camera in joint and 3D space. The algorithm optimizes the choice of a position where the viewing angle is acceptable, the number of occluded voxels is small and which is as close as possible to the previous camera position. These distributions over the viewpoints on the viewsphere sum to 1. They are used to normalize the different components of the objective function.

The goal is to choose parameters of a state s_{t+1} that maximize the following objective function:

$$\begin{aligned} P_{total}(s_{t+1} \mid s_t) = w_{vis}P_{vis}(s_{t+1}) + w_{nocc}P_{nocc}(s_{t+1}) + \\ + w_{dist}P_{dist}(s_{t+1} \mid s_t) + w_{jt}P_{jt}(s_{t+1} \mid s_t) \end{aligned} \tag{2}$$

where w_{vis}, w_{nocc}, w_{dist} and w_{jt} are the weights controlling the mixture of the distributions $P_{vis}(s_t)$, $P_{nocc}(s_{t+1})$, $P_{dist}(s_{t+1}|s_t)$, and $P_{jt}(s_{t+1}|s_t)$ that are responsible for visibility, occlusion and distance to travel in the 3D and joint spaces respectively.

The transition between the states is described with the variables of s_t and s_{t+1}. In the distributions we will include only variables on which they actually depend on to avoid notation overload.

The distribution that controls the angular distance traveled on the dome is based on the variable

$$\Delta\theta_d = \cos^{-1}(\vec{v}_t \cdot \vec{v}_{t+1}), \tag{3}$$

which corresponds to the angle between the viewpoints at states t and $t+1$.

The remaining part of this section presents the probability distributions that will account for the factors we are interested to control.

5.1. Viewpoint quality

The best overview of the scene is provided by the top viewpoint, which brings us to discourage solutions close to the boundary of the restricted dome (area within θ_{lim} view angle). All possible positions in the state $t+1$ that lie inside are given by the viewpoint angle θ_{t+1} in

$$\theta_{t+1} = \cos^{-1}(\vec{n} \cdot \vec{v}_{t+1}) \leq \theta_{lim}, \tag{4}$$

where the limit angle θ_{lim} between $\vec{n}$ and $\vec{v}_b$ is given by

$$\theta_{lim} = \cos^{-1}(\vec{n} \cdot \vec{v}_b). \tag{5}$$

It is used to scale the variables that control distance and angle. Thus the following distribution is used:

$$P_{vis}(s_{t+1}) = P_{vis}(\theta_{t+1}) = \frac{1}{Z_{vis}} e^{-\frac{1}{2}\left(\frac{2\theta_{t+1}}{\theta_{lim}}\right)^2}, \tag{6}$$

where $0 \leq \theta_{t+1} \leq \theta_{lim}$ and Z_{vis} is a normalization factor for the distribution. This formula favors angles less than $\frac{1}{2}\theta_{lim}$.

5.2. Target occlusion

Recall that $\vec{m}_{t+1}$ is the vector that contains number of projected occluding voxels for each camera position i as described in Sec. 4.2. The $P_{nocc}(s_{t+1})$ models whether a viewpoint has too much occlusion or not. If the number of occlusions seen from the camera position exceeds a specified threshold, the viewpoint is classified as occluded and $P_{nocc}(s_{t+1}) = 0$ for that particular viewpoint, otherwise $P_{nocc}(s_{t+1}) = 1$ marks a good viewpoint. Then

$$P_{nocc}(s_{t+1}) = \begin{cases} 1 & \text{if } m^i_{t+1} < m_0, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where m_0 is a threshold in the number of occluding voxels above which the viewpoint is considered occluded.

5.3. Travel distance

It is desirable to minimize the travel distance between two points that lie on the sphere so that the view does not change much. A distribution that will help us to minimize the distance between the current and predicted viewpoint is given below. The distance $d_{t+1,t}$ between points at states t and $t+1$ that lie on a sphere is given by

$$d_{t+1,t} = r\theta_{t+1,t}, \tag{8}$$

where r is the radius of the dome and $\theta_{t+1,t}$ can be computed by

$$\theta_{t+1,t} = \cos^{-1}(\vec{v}_{t+1} \cdot \vec{v}_t). \tag{9}$$

Using similar reasoning to the visibility distribution the distance distribution is:

$$P_{dist}(s_{t+1}|s_t) = P_{dist}(d_{t+1,t}) = \frac{1}{Z_{dist}} e^{-\frac{1}{2}\left(\frac{d_{t+1,t}}{f(\theta_{lim})}\right)^2}, \tag{10}$$

where $0 \leq d_{t+1,t} \leq 2\theta_{lim} r$. $f(\theta_{lim})$ and Z_{dist} are scaling factors for the random variable $d_{t+1,t}$ and the distribution respectively. In particular we set $f(\theta_{lim}) = \frac{\theta_{lim}}{2} r$. Since the radius of the dome is fixed P_{dist} can be simplified:

$$P_{dist}(d_{t+1,t}) = \frac{1}{Z_{dist}} e^{-\frac{1}{2}\left(\frac{2\theta_{t+1,t}}{\theta_{lim}}\right)^2}, \tag{11}$$

where $0 \leq \theta_{t+1,t} \leq 2\theta_{lim}$. The reason for scaling the random variable θ_d with $\frac{1}{2}\theta_{lim}$ is similar as before. In this case though, the scaling results in a much narrower distribution to favor smaller motions and thus better visual stabilization.

5.4. Robot arm joint parameters

As well as minimizing the sensor motion (see Sec. 5.3) it is also desirable to minimize the robot joint angle changes, so as to limit large changes in joint space configuration. This section gives the distribution for small transitions in the joint space between the set of joint parameters for the current and the predicted positions.

Let $\mathbb{J} \subseteq \mathbb{R}^n$ be the set of all allowed joint parameters that control the robot arm. For the robot arm we are using there are $n = 6$ parameters. Let $\vec{j}_t, \vec{j}_{t+1} \in \mathbb{J}$ be the joint parameters responsible for the current and predicted states of the robot arm respectively. Following a similar reasoning to the distance and angle the distribution $P_{jt}(s_{t+1}|s_t)$ is defined as:

$$P_{jt}\left(\vec{j}_{t+1} \mid \vec{j}_t, \Sigma\right) = \frac{1}{Z_{jt}} e^{-\frac{1}{2}(\vec{j}_{t+1}-\vec{j}_t)^\intercal \Sigma^{-1} (\vec{j}_{t+1}-\vec{j}_t)}, \tag{12}$$

where Z_{jt} is a normalization constant for the probability distribution. Furthermore, Σ is a diagonal approximation of the covariance matrix of the form $\Sigma = \sigma^2_{jt}\mathbb{I}$, where σ^2_{jt} is the variance of the distances in joint space between all camera positions.

6. Optimization

This section explains how the parameters of the objective function (2) are determined. In the first phase the cost weights $\vec{w}$ are trained to describe the desired behavior. This fully specifies the objective function, which can be in turn optimized to predict the next state at run-time.

6.1. Estimation of cost weights

To avoid having to manually assign ground truth predictions for a large number of viewpoints to train the cost weights, we instead prescribe the value of the cost function, which allows to describe the desired behavior more flexibly.

We use cross validation to train the weights $\vec{w}$ by minimizing a sum of squares distance between $P_{total}(s_{t+1}|s_t)$ and the prescribed function $\tilde{P}_{t+1,t}$ for all allowed states t and $t+1$. The score for each new state is given by the scoring function

$$\tilde{P}_{t+1,t} = \alpha_s \Omega_{t+1} + \alpha_d e^{-d_{t+1,t}} + \alpha_\theta e^{-\theta_t}, \tag{13}$$

where Ω_{t+1} evaluates to 1 when the viewpoint associated with s_{t+1} is not occluded (0 otherwise), the $d_{t+1,t}$ is the distance moved between the viewpoints and θ_t encourages a more vertical viewpoint.

Weights α_s, α_d and α_θ control the behavior of the objective function giving priority to either avoiding occlusion α_s, moving to closer viewpoints α_d or higher viewpoints α_θ. We use e^{-x} for both angle and distance because it takes larger values for smaller values of x and it drops exponentially for larger ones. Hence when the weights of P_{total} are trained with score $\tilde{P}_{t+1,t}$ it will pick neighboring viewpoints that will meet different criteria depending on the combination of the weights. In the experimental section, we explain how we choose the most appropriate set of weights.

To select the α weights of the objective function, we set a desired behavior which is controlled by the weights w of (13). To explore different behaviors of P_{total} we initialize $[\alpha_s, \alpha_d, \alpha_\theta]$ with values from the set $\{0.5, 1, 1.5, 2\}$ in all possible combinations allowing repetition. Hence there are 64 possible sets of weights and we look for the values that best produce the smooth behavior we seek. To choose the most appropriate behavior we monitor the number of jumps the algorithm makes, the average distance traveled and the average increase in the z direction (we want to move upward for better viewpoints) when s_t differs from s_{t+1}.

Given the α values, we then find the weights of the objective function (2) by picking the set of weights $\vec{w} = [w_{nocc}, w_{vis}, w_{dist}, w_{jt}]$ that approximates the desired behavior. These weights $\vec{w}^*$ are those that optimize (14) on both training and testing sets:

$$\vec{w}^* = \arg\min_{\vec{w}} \sum_{i,j \in S} \left(P_{total}(s_i \mid s_j) - \tilde{P}_{i,j}\right)^2, \qquad (14)$$

where S is the set of considered viewpoints and i, j are all pairs of viewpoints corresponding to states $t+1$ and t respectively. To find $\vec{w}$ we initialized a set of weights ranging from 0 to 1 from a uniform distribution and normalize their sum to 1. Then we find the minimizer of (14) using Matlab's built in function `fmincon`.

We have observed that a good performance is achieved when $w_{vis} \ll w_{dist}$, which is the case of the values actually used in the experiments (Sec. 7.2).

6.2. Inference of the next state

With the weights cost determined as in the previous section we can optimize if and where the camera should move given the current state. The next state s^*_{t+1} is chosen to maximize the function (2) in

$$s^*_{t+1} = \arg\max_{s_{t+1}} P_{total}(s_{t+1} \mid s_t). \qquad (15)$$

In our case the discrete set of positions can be simply enumerated to find the state with maximum value.

7. Experiments

In this section we present the results for the sets of weights both for the scoring and objective function. We discuss the different effects that each set of weights for the scoring function produces and choose the most appropriate ones to train P_{total}. To fuse the data from multiple depth sensors we use OctoMap [3], which filters the noise occurring due to temporal instability, interference or calibration inaccuracies, and produces an octree voxel map.

7.1. Experimental setup

The following apparatus depicted in Fig. 2 is used:

- workcell with table and supporting structure,
- four Kinect V2 depth sensors facing the region of interest, calibrated extrinsically using a pattern,
- UR 10 robot arm with 6 DOF top-mounted above the center of the table,
- camera mounted on the end-point of the robot arm,
- controlling computer with i7-6700 @ 3.4 GHz CPU and 16Gb of RAM.

Position $v_{1,1,1}$ corresponds to the bottom left corner wrt. the observer and v_{n_x,n_y,n_z} to the opposite top corner of the workcell space. We initialize the dome of radius 0.7 meters with two subdivisions of the initial icosahedron and obtain $n_1 = 44$ allowed camera poses in the upper part of the dome (see Fig. 6).

The experiments cover 8 different scenarios, 6 acquired with the setup described above and 2 simulated cases. For the simulated cases we manually occluded camera viewpoints to create specific extreme situations. For training the objective function weights we use cross validation with a training set 50% of the eight scenarios, ie. we trained on $\binom{8}{4} = 70$ cross-validation splits.

To decide whether a viewpoint is occluded we experimentally set the occupancy threshold to $m_0 = 3$.

7.2. Experimental parameters

Since our goal is to record a sequence, we not only need the best viewpoint, but also to have reasonably stabilized images. In this setup we focus on the case $w_{vis} \ll w_{dist}$ because it fits our requirements. To have the smoothest possible video recording and joint parameter transitions we choose the following set of weights for the score

$$[\alpha_s, \alpha_d, \alpha_\theta] = [0.5, 1, 2] \qquad (16)$$

that minimizes the number of jumps and the average distance traveled, and jump in z direction for consecutive

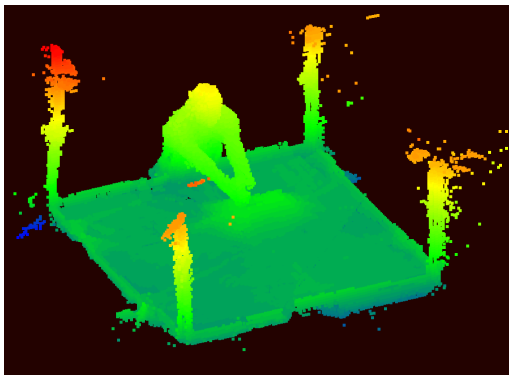


Figure 5: Example fused point cloud captured by the depth sensors, height colored.

states. Optimizing (14):

$$[w_{nocc}, w_{vis}, w_{dist}, w_{jt}] = [0.080, 0.238, 0.585, 0.096] \tag{17}$$

Using this set of weights results in average geodesic distance of 14.4 cm. The training took approximately 2.5 minutes using a single processor. To calculate NBV the algorithm requires 50 ms in the Matlab implementation.

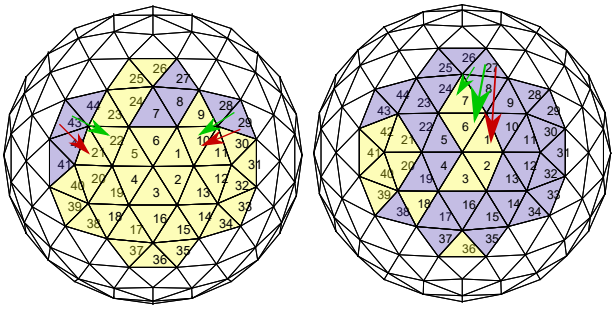
7.3. Evaluation

We have 8 different scenes represented by point clouds with different levels of occlusion. An example is given in Fig. 5. In each one we compared the output of our algorithm against the ground truth. The ground truth has been manually assigned, i.e. for each case a human supervisor has labeled the next best viewpoint. The limitation of the established ground truth is that the person cannot take into account joint space constraints. For each viewpoint in each scenario we estimate the best next view point regarding the occlusion, viewpoint, joint and geodesic distance. For each scenario there are 44 possible initial viewpoints (see Fig. 6) of which there are three non-reachable viewpoints (6,7,8) due to robot arm limits. We test the behavior of the algorithm for each case, as if every viewpoint would be the current state. The dataset with total $41 \times 8 = 328$ cases (of initial and next view) can be downloaded from the project website[2] .

There are 16 cases out of the 328 where the algorithm provided different output than the ground truth. For these 16 viewpoints we found out that either the position in the ground truth could not be reached because of the joint limits, or the joint configuration drastically changes because the arm has to flip over, i.e. large joint distance.

One example is given for Scn. 4 viewpoints 43 and 29. They are symmetric and occluded, as shown in Fig. 6a. The ground truth expected them to move to viewpoints 22 and 10 respectively (green arrow). However, our algorithm predicted the move from 43 to 21 instead (red arrow), due to the joint distance, while viewpoint 29 moves to 10 as expected. Another interesting case is in Scn. 7 (Fig. 6b). If we initialize from point 27, the algorithm results in viewpoint 1 instead of 6 or 7. This situation occurs because viewpoint 6 and 7 cannot be reached due to joint limits.

[2] http://homepages.inf.ed.ac.uk/rtylecek/DNBV



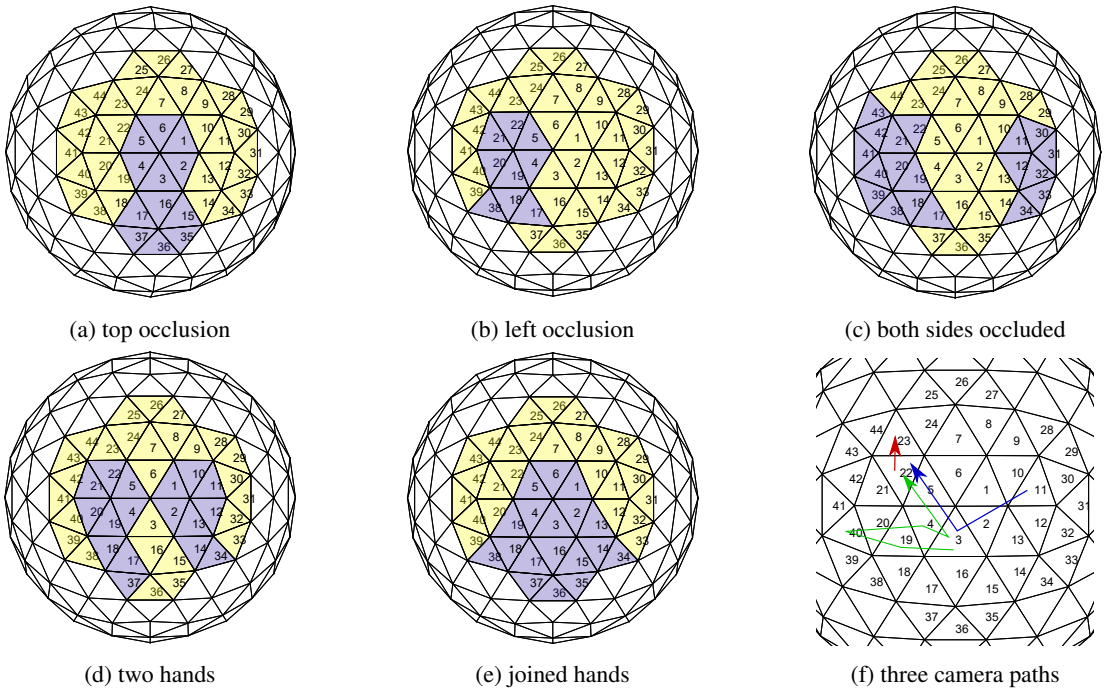
(a) scene 4 (b) scene 7

Figure 6: Enumeration of the 44 viewpoints on the dome. The numbers are only on the allowed camera viewpoints. Color indicates occlusion of viewpoints (purple: occluded, yellow: not occluded). Green and red arrows indicate the new position according to GT and algorithm prediction respectively (see text).

Scn.	Start	End	Distance	Joints	Angle
4	43	21	0.30	0.17	0.44
4	43	22	0.36	0.24	0.35
7	27	1	0.54	1.87	0.17
7	27	6	0.48	-	0.17
7	27	7	0.31	-	0.36

Table 1: Analysis of points that do not match the ground truth in the two example scenarios (Scn. 4 and 7). The columns above show the initial viewpoint, the final viewpoint, the geodesic and joint distance traveled from initial to final view point and viewpoint angle respectively. The '-' denotes points that are not reachable due to robot joint limits.

Using Table 1 we can justify the outputs of the algorithm. In the case where the algorithm decides to jump from viewpoint 43 to 21 it is clear that 21 beats 22 in all criteria but angle. But we consider the objective function weights that give higher priority to distance to have a smoother video. When the camera is initialized in viewpoint 27 in point cloud 7, viewpoints 6 and 7 are selected, but they are not reachable because of robot joint limits (see Fig. 6b).



(a) top occlusion (b) left occlusion (c) both sides occluded

(d) two hands (e) joined hands (f) three camera paths

Figure 7: Dynamic scene with moving obstacles. (a-e) show the occlusion projected on the dome (purple: occluded viewpoints, yellow: not occluded). (f) shows the camera trajectories for different starting positions.

7.4. Dynamic experiments

In this section we show the performance of the proposed algorithm in a dynamic scene. Since the faces in the dome are large, the occlusion does not change much between consecutive frames. Hence, we have chosen 5 key frames where the occlusion changes significantly to see the result of the method.

The occlusion sequence (Fig. 7) has an initial position where the views at the top of the dome are occluded (a), then the occlusion moves to the left (b), then another occlusion appears for views on on the other side of the dome (c). Later the two occluded regions go to the top of the dome (d) and finally they merge (e). This sequence is the case like an arm covering the top of the target point, moves to one side, then the other arm appears on the other side, and finally both arms move and overlap in the center of the scene.

For this experiment we choose 3 initial positions, corresponding to viewpoints 11 (not occluded), 3 (occluded) and 22 (not occluded). The results of the algorithm from the three starting positions (11→blue, 3→green, 22→red) are depicted in Fig. 7f, which illustrates the sequence of viewpoints given the changing data.

With a different configuration of parameters where the viewpoint weight is higher, the sequences of movements would have more changes of viewpoint as the algorithm tries to optimize the viewpoint. The dynamic behavior of the proposed method can be also observed in the video presented online[3].

8. Conclusion

This paper presented both a new problem of dynamically selecting viewpoints for a moving sensor in an environment with static and dynamic obstacles, and provided an algorithm to solve this problem. We proposed the use of a set of tessellations of a virtual dome as the set of possible viewpoints. The best next viewpoint becomes the one that minimizes four factors, occlusion, quality of viewpoint in terms of angle, geodesic distance and joint distance of the robot arm. We have evaluated our proposal in different scenarios showing good performance in all cases.

As future work, we want to explore more parameter configurations which will provide different behavior. Moreover, we are interested in including a smoothing factor to avoid rapid changes, and a prediction criterion in order to minimize jumps.

Acknowledgements

Authors acknowledge the support of EU project TrimBot2020 and project from University of Alicante (Gre16-28).

[3] https://youtu.be/v1vy_FdxCa4

References

[1] E. Dunn, J. van den Berg, and J.-M. Frahm. Developing visual sensing strategies through next best view planning. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4001–4008. IEEE, oct 2009. 2

[2] P. Gambadauro and A. Magos. Surgical videos for accident analysis, performance improvement, and complication prevention. *Surgical Innovation*, 19(1):76–80, 2012. 1

[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. 4, 6

[4] A. E. Hunt and A. C. Sanderson. Vision-Based Predictive Robotic Tracking of a Moving Target. Technical report, Carnegie-Mellon University, Robotics Institute., 1982. 2

[5] M. Karaszewski, M. Stępień, and R. Sitnik. Two-stage automated measurement process for high-resolution 3D digitization of unknown objects. *Applied Optics*, 55(29):8162, oct 2016. 2

[6] N. A. Massios and R. B. Fisher. A Best Next View Selection Algorithm Incorporating a Quality Criterion. In *Procedings of the British Machine Vision Conference 1998*, pages 78.1–78.10. British Machine Vision Association, 1998. 1, 2, 3

[7] R. Monica and J. Aleotti. Contour-based next-best view planning from point cloud segmentation of unknown objects. *Autonomous Robots*, feb 2017. 2

[8] K. Morooka, Hongbin Zha, and T. Hasegawa. Next best viewpoint (NBV) planning for active object modeling based on a learning-by-showing approach. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, volume 1, pages 677–681. IEEE Comput. Soc, 1998. 2

[9] N. Papanikolopoulos, P. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9(1):14–35, 1993. 2

[10] R. Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1016–1030, 1999. 2

[11] J. Sanchiz and R. Fisher. A Next-Best-View Algorithm for 3D Scene Recovery with 5 Degreesof Freedom. In *Procedings of the British Machine Vision Conference 1999*, pages 17.1–17.10. British Machine Vision Association, 1999. 1, 2, 4

[12] M. K. Singh, K. S. Venkatesh, and A. Dutta. A new next best view method for 3D modeling of unknown objects. In *2015 Third International Conference on Image Information Processing (ICIIP)*, pages 516–519. IEEE, dec 2015. 2

[13] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian. Volumetric Next-best-view Planning for 3D Object Reconstruction with Positioning Error. *International Journal of Advanced Robotic Systems*, 11(10):159, oct 2014. 2

[14] S. Zhang, Y. Miao, X. Li, H. He, Y. Sang, and X. Du. Determining next best view based on occlusion information in a single depth image of visual object. *International Journal of Advanced Robotic Systems*, 14(1):172988141668567, feb 2017. 2